REMARKS

In the Office Action, the Examiner rejected Claims 31-34, which were all of the then pending claims, under 35 U.S.C. §103 as being unpatentable over the prior art, principally U.S Patent 6,078,744 (Wolczko, et al.). Specifically, Claims 31, 33 and 34 were rejected as being unpatentable over Wolczko, et al. in view of U.S. Patent 6,367,072 (Atkinson, et al.), and Claim 32 was rejected as being unpatentable over Wolczko, et al. in vie of U.S. Patent 6,317,872 (Gee, et al.). The Examiner also rejected Claim 33 under 35 U.S.C. §112 as indefinite, noting that the claim does not identify a step "b," and the Examiner observed that the trademark "Java" is used in the application and commented that the trademark should be capitalized and accompanied by the generic terminology.

Applicants are herein amending Claims 31-34 to better define the subject matter of these claims. Also, new Claims 35 and 36, which are dependent from Claim 31, are being added to describe preferred features.

In particular, Claim 33 is being amended to identify steps "a," "b," and "c." It is believed that this fully addresses the Examiner's rejection of Claim 33 under 35 U.S.C. §112, and the Examiner is respectfully requested to reconsider and to withdraw this rejection of Claim 33.

Also, Applicants Attorneys have reviewed the specification, and it is believed that the trademarks used in the specification are properly used. For instance, on page 12, lines 18 and 21 the term "Java Virtual Machine" is used, and on page 22, line 30, the term "Java language" is used. In these instances, the trademark is capitalized and is followed by the appropriate generic language.

With respect to the rejection of the claims over the prior art, Applicants wish to note that an important difference between this invention and the cited references is that those references do not address the specific problem that is solved by the present invention.

To elaborate, the present invention, generally, relates to a method and system to compile programs or components of a program in a mixed static and dynamic environment. As explained in the present application, dynamic compilation involves translating a program component to machine code at run-time, before executing that program component. Static compilation involves the process of translating in an off-line manner and generating one or more binary codes to be executed at run-time. With existing machine and procedures, there are a number of problems with implementing dynamic and static compilation. These problems include performance overhead, testability and serviceability of compilation, dynamic binding and dynamic class loading.

The present invention effectively addresses these problems by providing a two step process. A compiler is used, in a first step, to perform one set of tasks, and then a virtual machine is used, in a second step, to perform additional tasks.

Specifically, Claim 31 is directed to a method for using a virtual machine to execute securely statically compiled code. In this method, a compiler performs a first set of integrity checks, and the virtual machine conducts a second set of integrity checks.

Claim 32 is directed to a procedure for linking precompiled code at run-time within a virtual machine. With this procedure, the compiler maintains certain symbolic entries, and the virtual machine uses these symbolic entries, before the code is executed, to generate direct references in the generated code.

Claim 33 is directed to a method for updating code at run time, when separately compiled code, which contains symbols, changes. With this method, the compiler generates certain code, and the virtual machine may be used to recompile that code under defined circumstances.

Claim 34 defines a method for maintaining compliance with a language requiring dynamic compliance, while still enabling the use of statically generated code for some byte code that depends on byte code that may be separately compiled. In accordance with this method, the compiler performs security features on byte code, and the virtual machine uses those security features to determine if the byte code has changed.

An important feature of the present invention, described in each of Claims 31-34, is the use of precompiled code.

More specifically, Claim 31 describes the feature that the method of the claim is for a virtual machine in which statically, precompiled code may be securely executed by a virtual machine; and Claim 32 describes the feature that the method defined by the claim is for linking separately statically, precompiled code at run-time. Also, Claim 33 is being amended to indicate that the method of the claim is for updating statically generated precompiled code at run time; and Claim 34 describes the feature that the method maintains full compliance with a language while enabling the use of statically generated precompiled code.

These claims all differ from Wolczko, et al. in that Wolczko does not disclose the use of precompiled code, as described in Claims 31-34. These claims describe the use of precompiled programs, not analysis information. The use of precompiled code allows for a better recompilation into native code.

The other references of record also fail to disclose or suggest this feature of the invention in the contexts of Claims 31-34.

For example, Atkinson, et al. describes a method and system to help ensure file integrity. This reference teaches incorporating a certification or signature in a file. This certification or signature may be confirmed at the recipient computer.

Gee, et al. was cited for its disclosure of a computer architecture for resolving symbolic references in code written in an object oriented programming language.

Neither Atkinson, et al. nor Gee, et al, though suggest the use of precompiled code for the purposes described in Claims 31-34.

This feature of the invention is of utility because it helps to achieve the advantages of combined static and dynamic compilation. The invention effectively achieves the reduced performance overhead of dynamic compilers while still achieving the aggressiveness that can be achieved with static compilers.

In light of the above-discussed differences between Claims 31-34 and the prior art, and because of the advantages associated with those differences, it cannot be said that any of Claims 31-34 would have been obvious to one of ordinary skill in the art. Accordingly, Claims 31-34 patentably distinguish over the prior art and are allowable. New Claims 35 and 36 are dependent from Claim 31 and are allowable therewith.

The Examiner is, thus, respectfully requested to reconsider and to withdraw the rejections of Claims 31-34 under 35 U.S.C. §103, and to allow these claims and new Claims 35 and 36.

Every effort has been made to place this case in condition for allowance, a notice of which is requested. If the Examiner believes that a telephone conference with Applicant' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

John S. Sensny

John S. Sensny
Registration No. 28,757
Attorney for Applicants

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343

JSS:jy